

Pseudo-random Sequences Generated by Cellular Automata

Bruno Martin and Patrick Solé*

*13S, Université de Nice–Sophia Antipolis, CNRS,
2000 route des Lucioles, BP 121, F-06903 Sophia Antipolis Cedex
{Bruno.Martin|Patrick.Sole}@unice.fr*

Abstract

Generation of pseudo random sequences by cellular automata, as well as by hybrid cellular automata is surveyed. An application to the fast evaluation and FPGA implementation of some classes of boolean functions is sketched out.

Introduction

Cellular Automata (CA) is a popular model of finite state machine with some pretention to generality and universality. Pseudo Random Sequences (PRS) on the other hand, have a long history of applications to computational (Monte Carlo sampling, numerical simulation) and communications problems (coding theory, streamciphers). In that context the popular model is the Linear Feedback Shift Register (LFSR), another model of linear finite state machine.

In the present work we survey the known attempts to generate PRS by CA. We give an account of the synthesis of LFSR by arrays of variable CA (known as hybrid CA or HCA). We sketch an application to the evaluation of boolean functions in n variables which are related to cyclic codes of length $2^n - 1$. This is aimed at VLSI implementation, especially by programmable arrays.

The material is organized as follows. Section 1 collects definitions and basic notions on PRS, CA and HCA. Section 2 reviews the synthesis theory of LFSR by HCA. Section 3 surveys the generation of PRS by elementary CA. Section 4 surveys the generation of PRS by HCA. Section 5 contains the application of synthesis theory to boolean functions evaluation.

1 Notations and definitions

1.1 (Pseudo-)randomness

This section recalls the classical definitions of pseudo-randomness. We first give an intuitive statement which gives the difference between real randomness and pseudo-randomness. We then introduce more formal definitions of pseudo-randomness.

* This work was supported by the french ANR program NUGET.

In [20], Wolfram describes three mechanisms responsible for random behavior in systems: (1) *Randomness from physics* like brownian motion; (2) *Randomness from the initial conditions* which is studied by chaos theory; and (3) *Randomness by design*, also called pseudo-randomness used in pseudo-random sequences generators. Many algorithms generate pseudo-random sequences. The behavior of the system is fully determined by knowing the seed and the algorithm used. They are quicker methods than getting "true" randomness from the environment, inaccessible for computers.

The applications of randomness have led to many different methods for generating random data. These methods may vary as to how unpredictable or statistically random they are, and how quickly they can generate random sequences. Before the advent of computational PRS, generating large amount of sufficiently random numbers (important in statistics and physical experimentation) required a lot of work. Results would sometimes be collected and distributed as random number tables or even CD iso-images.

More formally, a pseudo-random sequence (PRS for short) can be defined as:

Definition 1. *A sequence is pseudo-random if it cannot be distinguished from a truly random sequence by any efficient (polynomial time) procedure or circuit.*

Theorem 1 ([2]). *A sequence is pseudo-random iff it is next-bit unpredictable.*

Theorem 1 claims that for pseudo-random sequences, even if we know all the history, we don't have any information on the next bit. Theorem 1 was proved equivalent to:

Theorem 2 ([22]). *A PRS generator G passes Yao's test if, for any family of circuits F with a polynomial number of gates for computing a statistical test, G passes F .*

1.2 Cellular automata

In this section, we recall several definitions of cellular automata (CA). We focus on *elementary* cellular automata rules which restrict the set of the states to be \mathbb{F}_2 . A *cellular automaton* is generally a bi-infinite array of identical cells which evolve synchronously and in parallel according to a local transition function. The cells can only communicate with their nearest neighbors. Here, we will concentrate on two finite restrictions of CA:

- cyclic: a ring of N cells indexed by \mathbb{Z}_N .
- null boundary: an array of N cells in which both extremal cells are fed with zeroes.

All the cells are finite state machines with a finite number of states and a transition function which gives the new state of a cell according to its current state and the current states of its nearest neighbors.

Definition 2. *A cellular automaton is a finite array of cells. Each cell is a finite state machine $C = (Q, f)$ where Q is a finite set of states and f a mapping $f : Q^3 \rightarrow Q$.*

The mapping f , called *local transition function*, has the following meaning: the state of cell i at time $t + 1$ (denoted by x_i^{t+1}) depends upon the state of cells $i - 1$, i and $i + 1$ at time t (the *neighborhood* of cell i of radius 1). Fig. 1 illustrates one transition of a cellular automaton with 8 cells. The following equality rules the dynamics of the cellular automaton:

$$x_i^{t+1} = f(x_{i-1}^t, x_i^t, x_{i+1}^t) \quad (1)$$

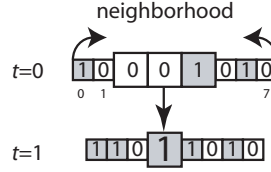


Fig. 1. Transition of a cell (rule 30); cyclic CA.

For a fixed t , the sequence of all the values x_i^t for $i \in \mathbb{Z}_N$, is the *configuration* at time t . It is a mapping c which assigns a state of Q to each cell of the cellular automaton. The sequence of configurations as pictured by Fig. 2 is called a *time-space* diagram. Fig. 2 depicts the evolution of a ring with $N = 8$ cells. On the top of Fig. 2, we have depicted rule 30 with each transition illustrated by three adjacent squares representing the different preimages of f and on the bottom, their image by f . A 0 (resp. 1) is painted white (resp. black). On the bottom of Fig. 2, we see the time-space diagram of the cellular automaton from the *initial configuration* at time $t = 0$ to time $t = 7$.

2 LFSR synthesis by HCA

We will restrict ourselves to the case where $Q = \mathbb{F}_2$ and f is a Boolean predicate with 3 variables, an *elementary rule*. These CA have been considered in [19]: there are 256 different binary CA and a natural number can be associated to each rule as follows:

$x_{i-1}^t x_i^t x_{i+1}^t$	111	110	101	100	011	010	001	000
x_i^{t+1}	0	0	0	1	1	1	1	0

The top line gives all possible preimages for f and the bottom line the images by f . Thus, f is fully specified by the 8-bit number written on the bottom line (00011110 in our example) which can be translated in basis 10 and then called the *rule* of the cellular automaton (as rule number 30 here). Equivalently, this rule can be considered as a Boolean function with (at most) 3 variables. Taking rule 30 again, its corresponding Boolean function is: $x_i^{t+1} = x_{i-1}^t \oplus (x_i^t \vee x_{i+1}^t)$ with \oplus denoting the Boolean XOR function and \vee the classical Boolean OR function. Its equivalent formulation in \mathbb{F}_2 is: $x_i^{t+1} = (x_{i-1}^t + x_i^t + x_{i+1}^t + x_i^t x_{i+1}^t)$.

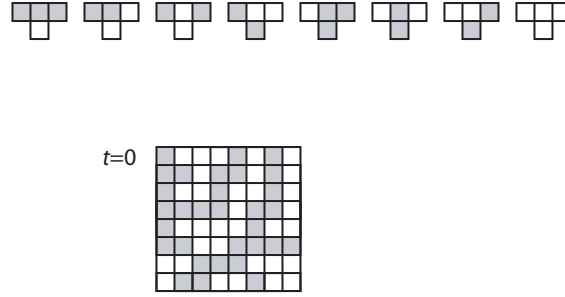


Fig. 2. Evolution of CA30 on a ring with $N = 8$ cells.

Equivalent rules Since we are dealing with pseudo-random generators, some of the elementary rules are equivalent by three transformations, all introduced by Wolfram in [19, p. 492]. We first introduce some notation: let us denote by \tilde{w} the mirror image of the finite binary word $w = w_1 \dots w_n$, $\tilde{w} = w_n \dots w_1$ and by \overline{w} the word obtained from w by exchanging the 0's by 1's (and conversely) $\overline{w} = \overline{w_1} \dots \overline{w_n}$. The first transformation is the *conjugation* which interchanges the roles of 0 and 1. It takes as an input r , the binary representation of a rule and returns \tilde{r} . For instance, the conjugation transforms rule 30 into rule 135. The second transformation, called *reflection* gives a re-ordering of the bits of r . Each bit $f_r(x_{i-1}, x_i, x_{i+1})$ is replaced by the value of $f_r(x_{i+1}, x_i, x_{i-1})$ (the mirror image of $x_{i-1}x_ix_{i+1}$) and leads to a re-ordering of the bits of r , the binary representation of the rule. As an example, by reflection, rule 30 is changed into rule 86. The last transformation combines boths and is called *conjugation-reflection*; it changes rule 30 into rule 149. All of these transformations keep the Walsh-Hadamard transform values of the cellular automata dynamics and are thus statistically equivalent.

2.1 Hybrid Cellular Automata

In the sequel, we will consider the case where different cells of the CA can use different rules. This model is called hybrid and will be denoted by HCA for short. In the context of sequence generation, several authors have considered this extension of the model of CA [4, 13, 15]. We will focus on *linear HCA* (LHCA) which is used by [4].

Linear hybrid cellular automata In [14, 4], Muzio et al. consider null-boundary hybrid CA which only use two rules: rule 90 and 150. In this case, a CA is fully specified by which cells use rule 90 and which use rule 150. This information is summarized in the *rule vector* $M = [d_0, d_1, \dots, d_{N-1}]$ such that $d_i = \begin{cases} 0 & \text{if cell } i \text{ uses rule 90} \\ 1 & \text{if cell } i \text{ uses rule 150} \end{cases}$.

Given M , its *reversal* is M 's mirror image: $[d_{N-1}, \dots, d_1, d_0]$. We also define the *sub-vector* $M_{i,j} = [d_i, \dots, d_j]$ with $i \leq j$ which also represents a submachine of the HCA consisting of cells i through j .

The encoding of rules 90 and 150 into zero and one, resp., means that equation (1) can be rewritten in \mathbb{F}_2 as $x_i^{t+1} = f_i(x_{i-1}^t, x_i^t, x_{i+1}^t) = x_{i-1}^t + d_i x_i^t + x_{i+1}^t$. We define the *state* of a HCA at time t to be the n -tuple formed from the state of the cells: $x^t = [x_0^t, x_1^t, \dots, x_{N-1}^t]^T$ (the superscript T denotes the transpose). Then, the next state function of the HCA is computed as $x^{t+1} = f(x^t)$. Since each f_i is linear, f is also linear and an endomorphism of \mathbb{F}_2^N . Linearity implies the existence of a matrix A such that $x^{t+1} = f(x^t) = A \cdot x^t$. The *HCA* transition matrix plays the same role as an LFSR transition matrix. A is tridiagonal.

$$A = \begin{pmatrix} d_0 & 1 & 0 & \dots & \dots & 0 & 0 \\ 1 & d_1 & 1 & \ddots & & & 0 \\ 0 & 1 & d_2 & \ddots & \ddots & & \vdots \\ \vdots & & & & 1 & d_{N-2} & 1 \\ 0 & 0 & \dots & \dots & 0 & 1 & d_{N-1} \end{pmatrix}$$

Let us denote by Δ the characteristic polynomial of A , that is $\Delta = |x\text{Id} - A|$.

Definition 3. [4] A polynomial p is said to be a *HCA polynomial* if it is the characteristic polynomial of some HCA.

Recall that $M_{i,j}$ is the HCA consisting of cells i through j and denote $\Delta_{i,j}$ its corresponding characteristic polynomial. When $i = 0$, we simply write M_k (resp. Δ_k) for the CA consisting of cells 0 to k (resp. its corresponding characteristic polynomial). Cattell and Muzio [4] proved that Δ_k satisfies a recurrence relation:

Theorem 3. [4] Δ_k satisfies the recurrence: $\Delta_{-2} = 0, \Delta_{-1} = 1, \Delta_k = (x + d_k)\Delta_{k-1} + \Delta_{k-2}$ for $k \geq 0$.

Theorem 3 provides an efficient algorithm to compute Δ_{N-1} the characteristic polynomial of a HCA from its rule vector M . Actually, this recurrence relation is related to Euclidean GCD algorithm on polynomials with Δ_k as the dividend, Δ_{k-1} as the divisor, $x + d_k$ as the quotient and Δ_{k-2} as the remainder. Applying Euclid's extended greatest division algorithm yields to the sequence of quotients whose constant terms are the mirror image of the rule vector. This comes from:

Lemma 1. [4] Let $p \in \mathbb{F}_2[x]$ and $q \in \mathbb{F}_2[x]$ of respective degrees n and $n - 1$. Then there exists a HCA with characteristic polynomial p and characteristic subpolynomial q if and only if applying Euclid's greatest division algorithm to p and q results in n degree one quotients.

Thus Δ_{N-1} and Δ_{N-2} determine the whole HCA. But in general, a characteristic polynomial isn't sufficient to uniquely determine the HCA. Just consider the following counter-example: $[0, 0, 1, 0, 0, 0] \leftrightarrow x^6 + x^5 + x^4 + x^3 + 1 \leftrightarrow [1, 1, 0, 1, 1, 1]$

To uniquely determine the HCA, we must know one more characteristic subpolynomial $\Delta_{1,N-1}$ and use theorem 4:

Theorem 4 (HCA quadratic congruence [4]). *Suppose we have a HCA with characteristic polynomial Δ_{N-1} and characteristic subpolynomials Δ_{N-2} and $\Delta_{1,N-1}$. Then both $y = \Delta_{N-2}$ and $y = \Delta_{1,N-1}$ satisfy the congruence: $y^2 + (x^2 + x)\Delta'_{N-1}y + 1 \equiv 0 \pmod{\Delta_{N-1}}$ where Δ'_{N-1} is the formal derivative of Δ_{N-1} in \mathbb{F}_2 .*

By combining Lemma 1 and Theorem 4, Cattell and Muzio give a characterization of HCA polynomials and give an algorithm for finding a HCA given a polynomial. Their method has been recently improved in [7].

Corollary 1. *Let $p \in \mathbb{F}_2[x]$ of degree n . Then p is a HCA polynomial if and only if for some solution q for y of the congruence*

$$y^2 + (x^2 + x)p'y + 1 \equiv 0 \pmod{p} \quad (2)$$

Euclid's greatest division algorithm on p and q results in n degree one quotients.

Theorem 4 has some weaknesses: it does not say neither that polynomials solutions to the quadratic congruence will be subpolynomials of Δ_{N-1} nor that non HCA polynomials won't have solutions to the quadratic congruence. Theorem 4 only gives necessary conditions for HCA polynomials: they have solutions to the quadratic congruence and that some of these solutions are subpolynomials. However, Theorem 4 is useful for irreducible polynomials:

Theorem 5. *If $p \in \mathbb{F}_2[x]$ is an irreducible polynomial of degree n , then equation (2) has exactly two solutions, both of which result in n degree one quotients.*

Corollary 2. *If $p \in \mathbb{F}_2[x]$ is an irreducible polynomial, then p has exactly two HCA realizations with one being the reversal of the other.*

Since one can build a HCA from an irreducible polynomial and represent it by its transition matrix, we can ask which is the relationship between LHCA and LFSR. If both are based on the same irreducible or primitive polynomial, they have the same behavior up to permutation of the order in which the states appear and the cycle structure of the states is identical. A similarity transform between LHCA and LFSR has been given in [5] and recently improved in [8].

3 PRS generation by CA

In [17, 18], Wolfram uses a one-dimensional cellular automaton for pseudo-random bit generation by selecting the values taken by a single cell when iterating the computation of rule 30 from an initial finite configuration where the cells are arranged on a ring of N cells. Mathematically, Wolfram claims the sequence $\{x_i^t\}_{t \geq 0}$ is pseudo-random for a given i . Wolfram extensively studied this particular rule, demonstrating its suitability as a high performance randomizer which can be efficiently implemented in parallel; indeed, this is one of the pseudo-random generators which was shipped with the connection machine CM2 and which is currently used in the Mathematica® software.

Unfortunately, this PRG is not suitable for cryptographic purpose. In [12], Meier and Staffelbach proposed a correlation attack to reverse the PRS generated by rule 30 although it passes classical statistical tests like the ones proposed in [9].

More recently, in [11], we have used a Walsh transform to explore the set of the 256 elementary rules. The Walsh transform is a well-known tool in the field of cryptology for studying the correlation-immunity of Boolean functions: Xiao and Massey [21] have characterized the notion of correlation-immunity with the Walsh transform. We have applied this technique to the pseudo-random sequences generated by all of the 256 binary rules and we provide evidence that there does not exist a non-linear rule which generates a correlation-immune pseudo-random sequence. Thus, we state Theorem 6.

Theorem 6. [11] *There is no non-linear correlation-immune elementary CA.*

And, according to Theorem 2, we can state that:

Corollary 3. *There is no elementary CA which can serve as PRS generator.*

So, does Theorem 6 annihilate any hope to design a good PRG by the means of CA? Not necessarily. Next section recalls the approach initiated by Tomassini and Sipper and section 5 describe another way of generating PRS with LHCA.

4 PRS generation by HCA

4.1 The cellular programming approach

Tomassini and Sipper [15] proposed to use HCA for generating better PRS. In this model, the rules are obtained by an evolutionary approach (a genetic algorithm). They have designed a *cellular programming* algorithm for cellular automata to perform computations, and have applied it to the evolution of pseudo-random sequence generators. Their genetic algorithm uses Koza's *entropy* $E_h = -\sum_{j=1}^{k^h} p_{h_j} \log_2 p_{h_j}$ where k denotes the number of possible values per sequence position, h a subsequence length and p_{h_j} is a measured probability of occurrence of a sequence h_j in a pseudo-random

sequence. It measures the entropy for the set of k^h probabilities of the k^h possible subsequences of length h . The entropy achieves its maximal value $E_h = h$ when the probabilities of the k^h possible sequences of length h are all equal to $1/\ell^h$, where ℓ^h denotes a number of possible states of each sequence. They have selected four rules of radius 1 for use in non-uniform cellular automata. The best rules selected by the genetic algorithm were rules 90, 105, 150 and 165 (which are all linear, a clear drawback).

A series of tests (including χ^2 test, serial correlation coefficient, entropy and Monte Carlo, but no correlation-immunity analysis) were made with good results, showing that co-evolving generators are at least as good as the best available CA randomizer. The authors also use elementary rules which we proved to be not correlation-immune. This was further investigated in [13].

Following the same kind of approach, Seredynski et al. in [13] have generalized the selection process to radius 2 rules. They use then both radius 1 and radius 2 rules in hybrid cellular automata. The rules selected by their genetic algorithm were 30, 86, 101 and 869020563, 1047380370, 1436194405, 1436965290, 1705400746, 1815843780, 2084275140 and 2592765285.

Their new set of rules was tested by a number of statistical tests required by the FIPS 140-2 standard [16] but no correlation-immunity analysis was made either.

4.2 The synthesis approach

This approach follows the synthesis algorithm proposed in [4]. They propose a method for the synthesis of a HCA from a given irreducible polynomial over \mathbb{F}_2 . The same problem for LFSR is well known as it can be directly obtained from the transition matrix. Furthermore, there is a one to one correspondence between LFSR's and polynomials. For CA, in general, a characteristic polynomial is not sufficient to uniquely determine the CA from which it was computed.

If we consider the characteristic polynomial Δ of the HCA (assumed to be irreducible), with α a root in \mathbb{F}_{2^n} . All n roots of Δ lie in \mathbb{F}_{2^n} . The roots $\alpha, \alpha^2, \alpha^{2^2}, \dots, \alpha^{2^{n-1}}$ are distinct and Δ can be factored in \mathbb{F}_{2^n} as $(x - \alpha)(x - \alpha^2)(x - \alpha^{2^2}) \dots (x - \alpha^{2^{n-1}})$.

Product of irreducibles Given p and q two irreducible characteristic polynomials and P and Q their respective transition matrix, one can build the transition matrix corresponding to $p \cdot q$. It can be defined by blocks as: $\begin{pmatrix} P & 0 \\ 0 & Q \end{pmatrix}$. This operation corresponds to the concatenation of LHCA [6]. They quoted that it permits to concatenate primitive machines for forming machines of much longer lengths.

5 Application: boolean functions evaluation

There is a well-known dictionary between, on the one hand, boolean functions in n variables, and binary sequences of period 2^n . More specifically, if f is such a function, and if we denote by \tilde{i} the base 2 expansion of i we can define a sequence by the rule $s_f(i) = f(\tilde{i})$, for $i \leq 2^n - 1$.

Many interesting boolean functions can be cast under the form $f(x) = Tr(ax + bx^s)$, where a, b are scalars of the extension field \mathbb{F}_{2^n} and Tr the trace function from \mathbb{F}_{2^n} down to \mathbb{F}_2 . In the case where n is odd and the Walsh Hadamard transform takes only three values they are the so-called *plateaued* boolean function of order $n - 1$ [23] also known as almost optimal or semi-bent.

They are the traces of so-called *almost bent AB* functions [3]. For monomials AB functions, the most famous exponents s are in the conjecturally exhaustive list of Gold, Kasami, Welch, Niho (see Table 1). In all these cases, an upshot of the theory of

Name	s	Condition
Gold	$2^i + 1$	$i \wedge m = 1, 1 \leq i \leq m/2$
Kasami	$2^{2i} - 2^i + 1$	$i \wedge m = 1, 1 \leq i \leq m/2$
Welch	$2^{(m-1)/2} + 3$	
Niho	$2^{2r} + 2^r - 1$	$r = t/2$ for t even $r = (3t + 1)/2$ for t odd with $1 \leq r \leq m = 2t + 1$

Table 1. Exponents of AB monomials.

Mattson-Solomon polynomials [10, p.249] is that the parity check polynomials of the attached cyclic codes (or, essentially, the connection polynomial of the LFSR) is of the form $m_\alpha m_{\alpha^s}$, where α generates \mathbb{F}_{2^n} over \mathbb{F}_2 . A fast algorithm to compute the minimal polynomials of elements in finite field extensions is given in [1].

6 Conclusion

We have used the synthesis approach to give an effective CA-realization of classical pseudo-random sequences of cryptographic quality. The main interest of this work would be to give an hardware implementation. The target hardware model of CAs is the Field Programmable Gate Arrays (known as FPGAs). FPGAs are now a popular implementation style for digital logic systems and subsystems. These devices consist of an array of uncommitted logic gates whose function and interconnection is determined by downloading information to the device. When the programming configuration is held in static RAM, the logic function implemented by those FPGAs can be dynamically re-configured in fractions of a second by rewriting the configuration memory contents. Thus, the use of FPGAs can speed up the computation done by the cellular automata. Putting all together allows high-rate pseudo-random generation of good quality.

References

1. D. Augot and P. Camion. On the computation of minimal polynomials, cyclic vectors, and frobenius forms. *Linear Algebra Appl.*, 260:61–94, 1997.
2. M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13:850–864, 1984.
3. C. Carlet, P. Charpin, and V. Zinoviev. Codes, bent functions and permutations suitable for DES-like cryptosystems. *Designs Codes and Cryptography*, 15:125–156, 1998.
4. K Cattell and J.C Muzio. Synthesis of one-dimensional linear hybrid cellular automata. *IEEE Trans. on Computer-aided design of integrated circuits and systems*, 15(3):325–335, 1996.
5. K Cattell and J.C Muzio. An explicit similarity transform between CA and LFSR matrices. *Finite fields and their applications*, 4:239–251, 1998.
6. K Cattell, S Zhang, X Sun, M Serra, J.C Muzio, and D.M Miller. One-dimensional LHCA: their synthesis, properties and applications in VLSI testing. Report, University of Victoria, B.C., Canada, 1998.
7. S.J Cho, U.S Choi, H.D Kim, Y.H Hwang, J.G Kim, and S.H Heo. New synthesis of one-dimensional 90/150 linear hybrid group CA. *IEEE Transactions on comput.-aided design of integrated circuits and systems*, 26(9):1720–1724, 2007.
8. D Kagaris. A similarity transform for linear finite state machines. *Discrete Applied Mathematics*, 154:1570–1577, 2006.
9. D.E. Knuth. *Seminumerical Algorithms*. Addison Wesley, 1969.
10. F.J. MacWilliams and N.J.A. Sloane. *The theory of error correcting codes*. North-Holland, 1977.
11. B. Martin. A Walsh exploration of elementary CA rules. *J. of Cellular Automata*, 2008. To appear.
12. W. Meier and O. Staffelbach. Analysis of pseudo random sequences generated by cellular automata. In *EUROCRYPT '91*, LNCS. Springer Verlag, 1991.
13. F. Seredynski, P. Bouvry, and A. Y. Zomaya. Cellular automata computations and secret key cryptography. *Parallel Comput.*, 30(5-6):753–766, 2004.
14. M. Serra, T. Slater, J.C. Muzio, and D.M. Miller. The analysis of one dimensional CA and their aliasing properties. *IEEE Trans. on Comput.-aided design*, 9:767–778, 1990.
15. M. Sipper and M. Tomassini. Co-evolving parallel random number generators. In *Parallel Problem Solving from Nature – PPSN IV*, pages 950–959, Berlin, 1996. Springer Verlag.
16. NIST FIPS publication 140-2, Security requirements for cryptographic modules. US Gov. Printing Office, Washington, 1997.
17. S. Wolfram. Cryptography with CA. In *CRYPTO 85*, LNCS. Springer Verlag, 1985.
18. S. Wolfram. Random sequence generation by cellular automata. *Adv. in applied math.*, 7:123–169, 1986.
19. S. Wolfram. *Theory and applications of cellular automata*. World Scientific, 1986.
20. S. Wolfram. *A new kind of science*. Wolfram Media Inc., Illinois, US, 2002.
21. G-Z. Xiao and J. L. Massey. A spectral characterization of correlation-immune combining functions. *IEEE Trans. on Information Theory*, 34(3):569–, 1988.
22. A.C. Yao. Computational information theory. In Y. Abu-Mostafa, editor, *Complexity in information theory*, pages 1–15. Springer Verlag New York, 1988.
23. Y. Zheng and X-M. Zhang. On plateaued functions. *IEEE Trans. Inform. Theory*, IT 47:1215–1223, 2001.